be desired. If the device 190 finds the GSB 612 reset, the device 190 places the current time-of-day value in the TOD register 670, and completes the I/O operation with only the completion vectors set 600.

[0042] Fig. 12 is a flow chart showing the tasks performed by the dispatcher 605 of the operating system above the line 699, and those tasks performed by each device 190, shown below the line 699. At 702, the dispatcher 605 initializes the system as previously described, which includes placing the current time-of-day value in the TOD register 670, and placing the target-delay-interval in the TDI register 672. At 704, the dispatcher 605 then begins to poll the hierarchy 600, as previously described, to locate devices that need attention.

[0043] At 706, during a send/receive I/O operation, a device 190 checks to determine if the GSB 612 is set. If the GSB 612 is set, a check is made at 708 to determine if the delay interval is exceeded. If the delay is exceeded at 708, the device adapter 191 drives a low level interrupt to the processor 700 of the host computer 210 without modifying the time-of-day value when the GSB was originally set, thus allowing for the full delay to be calculated when the GSB is finally reset. This interrupt is low cost because it only causes the processor 700 to poll the completion vectors. No detailed information is queued as to which device requires attention, therefore the amount of serialization / complexity required to drive the interrupt is significantly reduced. Since each device is assigned a unique lowest level completion vector byte, the control information describing the device can be easily obtained by maintaining a parallel vector of control block addresses. Specifically, once it is seen that completion vector byte 44 (for example) is set, that byte offset can be used as an index into an array of entries (each 4 or 8 bytes in length depending upon the addressing range), that contains the address of a control block that describes the device now known to be requiring attention. The interrupt also handles all devices needing attention at that time. Thus, interrupts from multiple sources are coalesced into a single notification event. This allows even the low cost interrupt to be amortized across multiple device completions.

[0044] If the GSB 612 is reset at 706, the device 190 sets the GSB 612, places the current time-of-day value in the TOD register 670 at 710 and completes the I/O operation with only the

completion vectors set 600. If the GSB is set at 706 but the delay is not exceeded at 708, then the I/O operation is completed at 714 with only the completion vectors 600 set. It will be understood that new TOD values and resetting the GSB occurs during the complete I/O step.

[0045] It will be understood that registering of I/O requests by each of the devices 190 in the hierarchy 600 is done independently from the polling of the hierarchy 600 by the dispatcher, and that the intelligent interrupt of the present invention is done by each device 190 in cooperation with but independent from the polling of the hierarchy 600 by the dispatcher 605.

[0046] If no completion occurs after the delay interval has been exceeded, but completions are pending, then a last resort timer is required to force the dispatcher 605 to perform the poll operation, even though it was never explicitly driven to do so.

[0047] The dispatcher 605 includes a program (Figs. 13 and 14) which calculates the TDI based on an algorithm which takes into account workload history. The overall model is to accumulate delay intervals (from time GSB is set to the time it is reset) over some number of samples. Once the threshold of samples has been reached, then the program makes a decision. The decision processing calculates the average interval since the last decision. If the average interval is within the target range, then the program requires some level of stability in getting good samples, before taking any action. If a single average interval is bad, then the program immediately zeros the delay interval, thereby resorting to interrupts only. The level of stability before setting a non-zero delay interval depends upon if the most recent last decisions to set a non-zero delay, turned out to be the wrong decision (i.e. delay probing non-zero delays if they haven't worked in the recent past)

[0048] The program processing of Dispatcher 605 is shown in Fig. 13 and is as follows:

800 Poll Global Summary Byte

802 If (set) Then

804 Calculate interval from time GSB was set to the Current TOD

805 Reset GSB 805

806 If (interval > BigThreshold) Then

808 Force a "bad" decision cycle (i.e. cause DelayInterval to go to zero, etc.)

End

810 Accumulate intervals across multiple samples

812 Increment the number of samples

814 If (# of samples is above a decision making threshold) Then

816 Call MakeDecision

End

End

[0049] The MakeDecision subroutine of Fig. 13 is shown in Fig. 14 and is as follows:

818 Save Probation indicator

820 Zero Probation indicator

822 Divide accumulated intervals by # of samples to obtain average interval

824 If (average > target threshold) Then

826 Zero GoalMet count

828 If (Saved probation is true) Then

830 Increment GoalMetMultiplier (capped at some value)

End

832 If (Current DelayInterval ^= 0) Then

834 Set DelayInterval to zero

End

Else (average is within target range)